Package 'basilisk.utils'

August 16, 2022

Version 1.8.0

Date 2021-04-08

Title Basilisk Installation Utilities

Imports utils, methods, tools, dir.expiry

Suggests knitr, rmarkdown, BiocStyle, testthat

biocViews Infrastructure

Description Implements utilities for installation of the basilisk package, primarily for creation of the underlying Conda instance. This allows us to avoid re-writing the same R code in both the configure script (for centrally administered R installations) and in the lazy installation mechanism (for distributed package binaries). It is highly unlikely that developers - or, heaven forbid, end-users! - will need to interact with this package directly; they should be using the basilisk package instead.

License GPL-3

RoxygenNote 7.1.1

VignetteBuilder knitr

git_url https://git.bioconductor.org/packages/basilisk.utils

git_branch RELEASE_3_15

git_last_commit 8a38d2e

git_last_commit_date 2022-04-26

Date/Publication 2022-08-16

Author Aaron Lun [aut, cre, cph]

Maintainer Aaron Lun <infinite.monkeys.with.keyboards@gmail.com>

R topics documented:

`activateEnvironment` *Activate a Conda environment*

Description

Mimic the (de)activation of a Conda environment by modifying environment variables in the current R process.

Usage

```
activateEnvironment(envpath = NULL, loc = getCondaDir())

deactivateEnvironment(listing)
```

Arguments

`envpath`	String containing the path to the Conda environment to activate. If `NULL`, the base Conda instance at `getCondaDir()` is activated.
`loc`	String containing the path to the root of a conda instance.
`listing`	Named list of strings containing name:value pairs for environment variables, typically the output of `activateEnvironment`.

Details

Conda environments generally need to be activated to function properly. This is especially relevant on Windows where the `"PATH"` variable needs to be modified for the DLL search. The `.activateEnvironment` function mimics the effect of activation by modifying environment variables in the current R session. This can be reversed by `.deactivateEnvironment` once the Conda environment is no longer in use.

The `.activateEnvironment` function will also unset a few bothersome environment variables:

- `"PYTHONPATH"`: to avoid compromising the version guarantees if **reticulate**'s `import` is allowed to search other locations beyond the specified Conda environment.

- "PYTHONNOUSERSITE": similarly, to avoid searching the user's site libraries.
- "RETICULATE_PYTHON": this would otherwise override any choice of Python, even after explicit specification via **reticulate**'s use_Condaenv!
- "RETICULATE_PYTHON_ENV": for similar reasons.

Value

activateEnvironment will modify environment variables to mimic activation of the Conda environment. It returns a named list of the previous values of all variables modified in this manner. (NA values indicate that the corresponding variable was not previously set.)

deactivateEnvironment restores the environment variables to their pre-activation state. It returns NULL invisibly.

Author(s)

Aaron Lun

Examples

```
# We can't actually run activateEnvironment() here, as it
# either relies on basilisk already being installed or
# it has a hard-coded path to the basilisk system dir.
print("dummy test to pass BiocCheck")
```

---

cleanConda *Run* conda clean

Description

Clean the Conda installation to remove unused packages and tarballs.

Usage

```
cleanConda(loc = getCondaDir())
```

Arguments

loc	String containing the path to the root of a conda instance.

Details

This should only be performed to save disk space for system installations, as the cached extras will never be used by any other **basilisk** package.

In contrast, for an externally cached conda, the extras may be re-used by other **basilisk** clients. So it's usually worth keeping them around, though this function can be called directly by the user if the cache gets too big.

Value

An integer value indicating whether the cleaning was successful.

Author(s)

Aaron Lun

Examples

```
# We can't actually run cleanConda() here, as it
# either relies on basilisk already being installed or
# it has a hard-coded path to the basilisk system dir.
print("dummy test to pass BiocCheck")
```

---

clearExternalDir *Clear the external installation directory*

Description

Clear the external installation directory by removing old Conda instances installed for different versions of **basilisk** with the same middle version number (i.e., same Bioconductor release).

Usage

```
clearExternalDir(path = getExternalDir())

clearObsoleteDir(path = getExternalDir())
```

Arguments

path	String containing the path to the latest version of the directory of interest.

Details

`clearObsoleteDir` can also be applied to the directories for the individual Conda environments, as the package version is also suffixed onto those directory paths. This is useful for clearing out obsolete versions of package environments.

Value

For `clearExternalDir`, all conda instances (and associated environments) of the same Bioconductor release as the current **basilisk** installation are destroyed.

The same applies for `clearObsoleteDir` except that the conda instance generated by the latest **basilisk** installation is retained.

Author(s)

Aaron Lun

See Also

getExternalDir, which determines the location of the external directory.

installConda, for the motivation behind this function.

Examples

```
# We can't actually run clearExternalDir() here, as it
# relies on basilisk already being installed.
print("dummy test to pass BiocCheck")
```

---

destroyOldVersions *Destroy old versions?*

Description

Should we destroy old installations of Conda from previous versions of **basilisk** (or old environment installations, for **basilisk** client packages)?

Usage

```
destroyOldVersions()
```

Details

The default value is TRUE, in order to save some hard drive space. This can be changed by setting BASILISK_NO_DESTROY environment variable to "1".

Value

Logical scalar providing an answer to the above.

Author(s)

Aaron Lun

See Also

installConda, where this function is used.

clearObsoleteDir, which may be triggered by this function.

`dir.create2` *Safe directory construction*

Description

Create a directory with an error message if it does not succeed.

Usage

```
dir.create2(path, recursive = TRUE, ...)
```

Arguments

`path, recursive, ...`
: Further arguments to pass to `dir.create`.

Details

This is primarily necessary to avoid incomprehensible errors when a directory cannot be created, usually due to insufficient permissions. We set `recursive=TRUE` by default for convenience.

Note that the presence of an existing directory at `path` will cause this function to fail. This is usually desirable in the context of **basilisk.utils** as stale directories should be `unlink2`ed beforehand.

Value

Either `path` is created or an error is raised. `NULL` is invisibly returned.

See Also

`unlink2`, for a similarly safe deletion function.

Examples

```
out <- tempfile()
dir.create2(out)
```

`getBinaries` *Get binary paths*

Description

Get binary paths

Usage

```
getCondaBinary(loc)

getPythonBinary(loc)
```

Arguments

`loc`	String containing the path to the root of a conda instance or environment.

Details

This code is largely copied from **reticulate**, and is only present here as they do not export these utilities for general consumption.

Value

String containing the path to the conda or Python executable inside `loc`. If `loc` is not supplied, the relative path from the root of the environment is returned.

Author(s)

Aaron Lun

Examples

```
getCondaBinary()

getPythonBinary()
```

getCondaDir *Get the* **basilisk** *Conda directory*

Description

Find the installation directory for the **basilisk**-managed Conda instance.

Usage

```
getCondaDir(installed = TRUE)
```

Arguments

installed	Logical scalar indicating whether **basilisk** is already installed.

Details

By default, conda is installed to a location specified by `getExternalDir`. This ensures that R package build systems do not attempt to generate binaries that include the conda instance; such binaries are not relocatable due to the presence of hard-coded paths, resulting in run-time failures.

If the `BASILISK_EXTERNAL_CONDA` environment variable is set to a path to an existing conda instance, the function will return it directly without modification. This allows users to use their own conda instances with **basilisk** but, in turn, they are responsible for managing it.

If the `BASILISK_USE_SYSTEM_DIR` environment variable is set to `"1"`, the function will return a path to a location inside the `basilisk` system installation directory. This is the ideal approach when installing from source as any conda and **basilisk** re-installations are synchronized. It also ensures that any R process that can load **basilisk** will also have permissions to access the conda instance, which makes life easier for sysadmins of clusters or other shared resources.

We suggest always calling this function after an `installConda` call, which guarantees the presence of the conda installation directory (or dies trying). Setting `installed=FALSE` should only happen inside the **basilisk** `configure` script.

Value

String containing the path to the conda instance.

Author(s)

Aaron Lun

Examples

```
# Setting the environment variable to run this example:
# all other modes rely on installation of basilisk.
old <- Sys.getenv("BASILISK_USE_SYSTEM_DIR")
Sys.setenv(BASILISK_USE_SYSTEM_DIR=1)
```

```
getCondaDir(installed=FALSE)

Sys.setenv(BASILISK_USE_SYSTEM_DIR=old)
```

---

getExternalDir *Get an external conda directory*

Description

Define an external location for installing the conda instance and **basilisk** environments.

Usage

```
getExternalDir()
```

Details

The default path contains the version number so that re-installation of **basilisk** will install a new instance of Conda. (This assumes that **basilisk** and **basilisk.utils** have synchronized version bumps.) See `installConda` for more details on how old versions of Conda are managed in this external directory.

If the `BASILISK_EXTERNAL_DIR` environment variable is set to some location, this will be used instead as the installation directory. Setting this variable is occasionally necessary if the default path returned by `R_user_dir` has spaces; or on Windows, if the 260 character limit is exceeded after combining the default path with deeply nested conda paths.

We assume that the user has read-write access to the external directory. Write access is necessary to generate new environments and to handle locking in `lockExternalDir`.

Value

String containing a path to an appropriate external folder. The last component of the path will always be the **basilisk** version number.

Author(s)

Aaron Lun

See Also

`getCondaDir`, to obtain the Conda installation directory.

Examples

```
# We can't actually run getExternalDir() here, as it
# either relies on basilisk already being installed.
print("dummy test to pass BiocCheck")
```

getSystemDir *Get the system installation directory*

Description

Get the system installation directory for a package. This is not entirely trivial as it may be called before the package is installed.

Usage

```
getSystemDir(pkgname, installed)
```

Arguments

pkgname	String containing the package name.
installed	Logical scalar specifying whether the package is likely to be installed yet.

Value

String containing the path to the (likely, if `installed=FALSE`) installation directory for `pkgname`.

Author(s)

Aaron Lun

Examples

```
getSystemDir("basilisk", installed=FALSE)
```

installConda *Install (Mini)conda*

Description

Install conda - specifically Miniconda, though historically we used Anaconda - to an appropriate destination path, skipping the installation if said path already exists.

Usage

```
installConda(installed = TRUE)
```

Arguments

installed	Logical scalar indicating whether **basilisk** is already installed. Should only be set to `FALSE` in **basilisk** `configure` scripts.

Details

This function was originally created from code in https://github.com/hafen/rminiconda, also borrowing code from **reticulate**'s `install_miniconda` for correct Windows installation. It downloads and runs a Miniconda installer to create a dedicated Conda instance that is managed by **basilisk**, separate from other instances that might be available on the system. Currently, we use version 4.8.3 of the Miniconda3 installer.

The installer itself is cached to avoid re-downloading it when, e.g., re-installing **basilisk** across separate R sessions. Users can obtain/delete the cached installer by looking at the contents of the parent directory of `getExternalDir`. This caching behavior is disabled for system installations (see `useSystemDir`), which touch nothing except the system directories; in such cases, only repeated installation attempts in the same R session will re-use the same installer.

Value

A conda instance is created at the location specified by `getCondaDir`. Nothing is performed if a complete instance already exists at that location. A logical scalar is returned indicating whether a new instance was created.

Destruction of old instances

Whenever `installConda` is re-run (and `BASILISK_USE_SYSTEM_DIR` is not set, see `?getCondaDir`), any previous conda instances and their associated **basilisk** environments are destroyed. This avoids duplication of large conda instances after their obselescence. Client packages are expected to recreate their environments in the latest conda instance.

Users can disable this destruction by setting the `BASILISK_NO_DESTROY` environment variable to `"1"`. This may be necessary on rare occasions when running multiple R instances on the same Bioconductor release. Note that setting this variable is not required for R instances using different Bioconductor releases; the destruction is smart enough to only remove conda instances generated from the same release.

Author(s)

Aaron Lun

Examples

```
# We can't actually run installConda() here, as it
# either relies on basilisk already being installed or
# it has a hard-coded path to the basilisk system dir.
print("dummy test to pass BiocCheck")
```

isWindows *Find the operating system*

Description

Indicate whether we are on Windows or MacOSX.

Usage

```
isWindows()

isMacOSX()
```

Value

Logical scalar indicating whether we are on the specified OS.

Author(s)

Aaron Lun

Examples

```
isWindows()
isMacOSX()
```

lockExternalDir *Lock external directory*

Description

Lock the external Conda installation directory so that multiple processes cannot try to install at the same time.

Usage

```
lockExternalDir(path = getExternalDir(), ...)

unlockExternalDir(lock.info, ...)
```

Arguments

path	String containing the path to the external directory.
...	For lockExternalDir, further arguments to pass to lockDirectory such as exclusive. For unlockExternalDir, further arguments to pass to unlockDirectory such as clear.
lock.info	A lock object generated by lockDirectory.

Details

This will apply a lock to the (possibly user-specified) external Conda installation directory, so that a user trying to run parallel **basilisk** processes will not have race conditions during lazy Conda installation. We use **dir.expiry** to manage the locking process for us, with the following strategy:

- If a system installation is being performed, we do not perform any locking. Rather, the R package manager will lock the entire R installation directory for us.
- If the external directory is not yet present, we establish an exclusive lock. We then proceed to the creation of said directory and installation of Conda.
- If an external installation directory is already present, we establish a shared lock. This will wait for any exclusive lock to expire (and thus any currently running installation to finish). No waiting is required if there are no existing exclusive locks.

Note that locking is only required during installation of Conda (or its environments), not during actual use. Once an installation/environment is created, we assume that it is read-only for all processes. Technically, this might not be true if one were to install a new version of **basilisk** halfway through an R session, which would prompt `installConda` to wipe out the old Conda installations; but one cannot in general guarantee the behavior of running R sessions when package versions change anyway, so we won't bother to protect against that.

Value

`lockExternalDir` will return a lock object from `lockDirectory`.

`unlockExternalDir` will unlock the file and return `NULL` invisibly.

Author(s)

Aaron Lun

See Also

`installConda`, for an example of how to implement this locking approach.

Examples

```
loc <- lockExternalDir()
unlockExternalDir(loc)
```

---

setCondaPackageDir *Set or unset the Conda package directory*

Description

Set or unset the directory used to store the cached Conda packages, e.g., tarballs and such. This should be a non-temporary location as other packages may link to its contents.

Usage

```
setCondaPackageDir(loc)
```

Arguments

loc	A string containing a path to the desired directory (that should already exist). Alternatively NA, in which case any existing setting is removed.

Value

The previous value of CONDA_PKGS_DIRS, invisibly.

Author(s)

Aaron Lun

Examples

```
# Setting it to something new:
out <- setCondaPackageDir(tempdir())

# Setting it back
setCondaPackageDir(out)
```

---

setVariable *Set an environment variable*

Description

Set an environment variable safely, unsetting it if the supplied value is NA.

Usage

```
setVariable(name, value)
```

Arguments

name	String containing the name of an environment variable.
value	String containing the value of an environment variable. This can be NA to unset the variable.

Value

String containing the value of the variable before running this function; or NA, if the variable was not set.

Author(s)

Aaron Lun

`unlink2` *Safe file deletion*

Description

Delete files or directories with an error message if it does not succeed.

Usage

```
unlink2(x, recursive = TRUE, force = TRUE, ...)
```

Arguments

`x, recursive, force, ...`
: Further arguments to pass to `unlink`.

Details

This is primarily necessary to avoid incomprehensible errors when a directory containing a stale environment or installation is not successfully deleted. We set `recursive=TRUE` by default for convenience; we also set `force=TRUE` by default to avoid difficulties due to rogue permissions.

Value

Either all `x` are successfully deleted or an error is raised. `NULL` is invisibly returned.

See Also

`dir.create2`, for a similarly safe directory creation function.

Examples

```
out <- tempfile()
unlink2(out) # no error from deleting non-existent file.

write(file=out, "whee")
unlink2(out)
```

useSystemDir *Use the R system directory?*

Description

Should we use the R system directory for installing **basilisk**'s Conda instance (or client environments)?

Usage

```
useSystemDir()
```

Details

The default value is `FALSE` to avoid problems with position-dependent code in packaged binaries. This can be changed by setting `BASILISK_USE_SYSTEM_DIR` environment variable to `"1"`.

Value

Logical scalar providing an answer to the above.

Author(s)

Aaron Lun

See Also

`getCondaDir`, where this function is used.

Index